

16時より開始いたします。もうしばらくお待ちください

- 後日ご共有の為、本セミナーは録画させていただきます。
- SmartDB環境にログインしてお待ちください。

※Firefoxまたはchromeでの実施をお願いいたします。

- Zoomの操作についてはチャット機能でご質問ください。

送信先を「全員」にした状態でチャットしてください。

※宛先が全員でない場合投稿に気づけない場合がございます。



- 本日使用するPower Shellコードのご確認をお願いいたします。

【3/8】に送付している案内メールの添付資料に記載しております。



15:00より開始いたします。もうしばらくお待ちください。

スマラジ!

SmartDB Radio



活用レベルアップ!



SmartDB 深掘りセミナー



スマラジ! SmartDBのAPIを使ってみよう!
～全6回で習得! REST APIアプリ操作基本編②～

株式会社ドリーム・アーツ



協創パートナー推進本部
中川 嵩之

1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

1. APIとは、外部から**機能・情報**を利用するための**窓口**である
2. APIには**リクエスト**と**レスポンス**という2つの通信がある
3. REST APIでは、データの**取得**、**登録**、**更新**、**削除**ができる
4. リクエストは、主に4つの要素で構成される

要素	説明
URI	: 処理を行う対象のリソース
HTTPメソッド	: 実行する処理内容
リクエストボディ	: 登録、更新するデータの詳細
リクエストヘッダ	: 認証などの補足情報

5. レスポンスは、主に2つの要素で構成される

要素	説明
ステータスコード	: 実行した結果
レスポンスボディ	: 返却されたデータの詳細



1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

STEP 01

バインダトークンの契約

STEP 02

バインダトークンの発行

STEP 03

バインダトークンの連携

STEP 04

実行プログラムの開発

バインダAPIトークンのオプションを契約

オプション	価格
バインダ追加	月額 100,000円/20バインダ
コラボレーター機能	月額 3,000円/10コラボレーター
バインダAPI	月額 100,000円/10トークン
ディスク容量追加	営業担当までご相談ください
ファイル一括アップロード	1,200,000円/1ライセンス ※購入から1年間の利用権利、期間終了時に失効

STEP 01

バインダトークンの契約

STEP 02

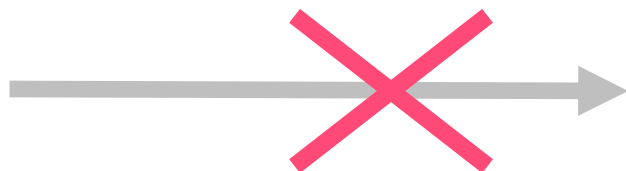
バインダトークンの発行

STEP 03

バインダトークンの連携

STEP 04

実行プログラムの開発



スマートデービー
SmartDB

第三者からの不正利用を防止するなど
APIの安全性を保つためにトークンを利用

STEP 01

バイндаトークンの契約

STEP 02

バイндаトークンの発行

STEP 03

バイндаトークンの連携

STEP 04

実行プログラムの開発

システム管理画面からバイндаトークンを発行

無効/有効
有効

名前	有効期限	先月検証回数	今月検証回数	登録バイнда
スマラジ！REST API基本編	2025/1/4	0	0	編集 バイндаを確認

表示件数: 10 1-1 / 1

[発行](#)

システム管理 > 認証とセキュリティ > バイндаトークン

名前*
01 ●●バイнда (制限なし)

有効期限
 無期限 有効期限を指定する

有効期限
2025/01/04

メモ
●●バイндаの外部連携で利用

※発行したAPIトークンは詳細画面で確認可能です

保存

STEP 01

バインダトークンの契約

STEP 02

バインダトークンの発行

STEP 03

バインダトークンの連携

STEP 04

実行プログラムの開発

対象のバインダにバインダトークンを紐づけ

バインダトークン

一覧 登録

トークンID * 1930a35d-1d37-43ad-9c06-33cdc2a6cb05

有効

権限 閲覧のみ 制限なし

説明文

一覧 登録

管理者用サイドメニュー > 外部連携 > トークン設定

< 一覧へ戻る

名前
01 ●● バインダ (制限なし)

有効期限
2025/1/4

バインダトークン
***** コピー

トークンID
1930a35d-1d37-43ad-9c06-33cdc2a6cb05 コピー

トークンタイプ
バインダトークン

メモ
●● バインダの外部連携で利用

システム管理 > 認証とセキュリティ > バインダトークン

STEP 01

バインダトークンの契約

STEP 02

バインダトークンの発行

STEP 03

バインダトークンの連携

STEP 04

実行プログラムの開発

SmartDBの情報・RestDocを参照してリクエストを作成

参照情報

ACCOUNT >

ROLE >

BINDER >

DOCUMENT >

FILE >

CATEGORY >

PROCESS >

LUXOR >

SESSION >

OTHERS >

SmartDB REST API (v3)

Download OpenAPI specification: [Download](#)

SmartDB ではHTTPアクセスで実行可能なREST API を利用することが可能です。

APIの実行形式

SmartDBのREST APIは、下記の形式で実行できます。

```
https://(SmartDBドメイン)/hibiki/rest/3/(対象API)
```

REST API実行時のセッションについて

REST APIの実行にはSmartDBのセッションが必要となります。
セッションの利用方法は下記の3パターンが考えられます。

- **バインダトークンを利用して実行する。 ※バインダAPIオプション限定**
管理画面の「バインダトークン」画面で発行したトークンを利用してREST APIを実行します。
実行時には後述の形式でHTTP Headerにトークンを指定してください。
バインダトークンを利用してAPIを実行する場合、下記の特徴を持ちます。
 - トークンを実行するためのロボットユーザとして実行されます。
※ロボットユーザのタイムゾーンは+0900
 - ParameterのcsrfTokenは不要です。
 - バインダを指定するAPIの場合、トークンを設定したバインダにのみ権限を持ちます。
※バインダ参照とlookupなどを利用する場合、参照先にもトークンを設定する必要があります。

HTTPヘッダーに以下を指定してください。

リクエスト

URI

HTTPメソッド

リクエストヘッダ

リクエストボディ

問題

SmartDBのREST API利用の流れとして正しいのは？

※ DCS / オンプレミスの場合、手順が異なる場合がございます。

1. SmartDB REST APIではトークンを利用する必要がない。
2. SmartDB REST APIでは第三者の不正利用を防止するためにトークンを利用する必要がある。
3. バインダトークンを発行後、業務プロセス定義とトークンを連携させる必要がある。
4. SmartDB REST APIのリクエストを作成するためにすべてのAPIを暗記する必要がある。

問題

SmartDBのREST API利用の流れとして正しいのは？

※ DCS / オンプレミスの場合、手順が異なる場合がございます。

1. SmartDB REST APIではトークンを利用する必要がない。
2. SmartDB REST APIでは第三者の不正利用を防止するためにトークンを利用する必要がある。
3. バインダトークンを発行後、業務プロセス定義とトークンを連携させる必要がある。

解説

1. SmartDB REST APIを利用する場合、セキュリティ向上のためにバインダトークンが必要である
2. バインダトークンは、REST APIを利用するバインダと連携する必要がある
3. リクエストはSmartDB環境情報や公開ドキュメントであるRetsDocを参照して作成できる

1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

BINDER

バインダー一覧取得
バインダ詳細情報取得
部品定義情報取得
よく使うバインダー一覧取得
よく使うバインダー一覧に追加
よく使うバインダー一覧から削除
ビュー一覧取得
ビュー詳細情報取得

DOCUMENT

文書詳細情報取得
文書新規登録
文書更新
文書削除
リスト型部品一覧取得
リスト型部品削除
リスト型部品の行新規登録
リスト型部品の行更新
リスト型部品の行削除
ブランチ文書取得
自動再計算の履歴リスト取得

自動再計算の履歴取得

指定したビューの文書一覧取得
指定したビューの文書詳細情報取得
部品値分布取得
文書一覧のCSV出力
CSV入力（新規）
CSV入力（更新）
コンプレックスCSV入力（新規）
リッチコメントリスト取得
リッチコメントの投稿
リッチコメントの詳細情報
リッチコメントの更新
リッチコメントの削除
リッチコメントのリアクション情報取得
リッチコメントのリアクション更新
文書一括操作
文書既読・未読ユーザ数取得
文書既読情報取得
文書未読情報取得
文書既読・未読フラグ取得
文書既読フラグ設定
バックグラウンドジョブステータス取得

PROCESS

開始可能業務一覧取得
よく使う業務一覧取得
よく使う業務一覧に追加
よく使う業務一覧から削除
業務プロセス開始
開始済み一覧取得
実施済一覧取得
履歴一覧取得
ワークリスト取得
実施履歴取得
アクティビティ詳細情報取得
アクティビティ実施
文書情報に基づいて実行中アクティビティ情報取得
指定ユーザのすべての代行設定取得
代行設定新規作成
代行設定取得
代行設定更新
代行設定削除
:
:

BINDER

- バインダー一覧取得
- バインダ詳細情報取得
- 部品定義情報取得
- よく使うバインダー一覧取得
- よく使うバインダー一覧に追加
- よく使うバインダー一覧から削除
- ビュー一覧取得
- ビュー詳細情報取得

DOCUMENT

- 文書詳細情報取得
- 文書新規登録
- 文書更新
- 文書削除
- リスト型
- リスト型
- リスト型部品新規登録
- リスト
- リス
- ブ
- 自
- 得



SmartDBのREST APIってたくさん種類がある・・・。
何をどんな風に使えばいいんだろう・・・？

- 自動再計算の履歴取得
- 指定したビューの文書一覧取得
- 指定したビューの文書詳細情報取得

PROCESS

- 開始可能業務一覧取得
- よく使う業務一覧取得
- に追加
- から削除

- リッチコメントの削除
- リッチコメントのリアクション情報取得
- リッチコメントのリアクション更新
- 文書一括操作
- 文書既読・未読ユーザ数取得
- 文書既読情報取得
- 文書未読情報取得
- 文書既読・未読フラグ取得
- 文書既読フラグ設定
- バックグラウンドジョブステータス取得

- 細情報取得
- アクティビティ実施
- 文書情報に基づいて実行中アクティビティ情報取得
- 指定ユーザのすべての代行設定取得
- 代行設定新規作成
- 代行設定取得
- 代行設定更新
- 代行設定削除
- ⋮
- ⋮

1. 前回の復習
2. SmartDBのAPIの始め方
3. よく使うSmartDB REST API
 - バインダ / 文書
 - 業務プロセス
4. ハンズオン
5. まとめ
6. 次回予告

BINDER

- バインダ一覧取得
- バインダ詳細情報取得
- 部品定義情報取得
- よく使うバインダ一覧取得
- よく使うバインダ一覧に追加
- よく使うバインダ一覧から削除
- ビュー一覧取得
- ビュー詳細情報取得

DOCUMENT

- 文書詳細情報取得
- 文書新規登録
- 文書更新
- 文書削除
- リスト型部品一覧取得
- リスト型部品削除

- リスト型部品の行新規登録
- リスト型部品の行更新
- リスト型部品の行削除
- ブランチ文書取得
- 自動再計算の履歴リスト取得
- 自動再計算の履歴取得
- 指定したビューの文書一覧取得
- 指定したビューの文書詳細情報取得
- 部品値分布取得
- 文書一覧のCSV出力
- CSV入力（新規）
- CSV入力（更新）
- コンプレックスCSV入力（新規）
- リッチコメントリスト取得
- リッチコメントの投稿
- リッチコメントの詳細情報
- リッチコメントの更新

- リッチコメントの削除
- リッチコメントのリアクション情報取得
- リッチコメントのリアクション更新
- 文書一括操作
- 文書既読・未読ユーザ数取得
- 文書既読情報取得
- 文書未読情報取得
- 文書既読・未読フラグ取得
- 文書既読フラグ設定
- バックグラウンドジョブステータス取得

BINDER

バインダ一覧取得
バインダ詳細情報取得
部品定義情報取得
よく使うバインダ一覧取得
よく使うバインダ一覧に追加
よく使うバインダ一覧から削除
ビュー一覧取得
ビュー詳細情報取得

DOCUMENT

文書詳細情報取得
文書新規登録
文書更新
文書削除
リスト型部品一覧取得
リスト型部品削除

リスト型部品の行新規登録
リスト型部品の行更新
リスト型部品の行削除
ブランチ文書取得
自動再計算の履歴リスト取得
自動再計算の履歴取得
指定したビューの文書一覧取得
指定したビューの文書詳細情報取得
部品値分布取得

文書一覧のCSV出力

CSV入力（新規）

CSV入力（更新）

コンプレックスCSV入力（新規）
リッチコメントリスト取得
リッチコメントの投稿
リッチコメントの詳細情報
リッチコメントの更新

リッチコメントの削除
リッチコメントのリアクション情報取得
リッチコメントのリアクション更新

文書一括操作

文書既読・未読ユーザ数取得
文書既読情報取得
文書未読情報取得
文書既読・未読フラグ取得
文書既読フラグ設定
バックグラウンドジョブステータス取得

PINDEX

リスト型部品の行新規登録

リッチコメントの削除

行更新

行削除

得

歴リスト

歴取得

の文書一

の文書計

出力

)

)

CSV入力

リスト取

の投稿

の詳細情

リッチコメントの更新

SmartDBの文書を他システムに連携

他システムのデータをSmartDBの文書に連携



スマートデービー
SmartDB

他システム



スマートデービー
SmartDB

他システム

リスト型部品削除

PINDB

リスト型部品の行新規登録

リッチコメントの削除

SmartDBの文書を他システムに連携

他システムのデータをSmartDBの文書に連携



行更新
行削除
得
歴リスト
歴取得
の文書一
の文書計

出力

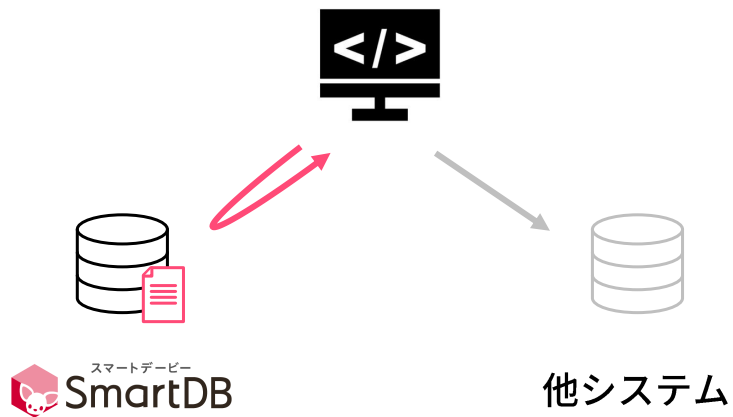
)
)
CSV入力
リスト取
の投稿
の詳細情

リッチコメントの更新

リスト型部品削除

SmartDBの文書を他システムに連携

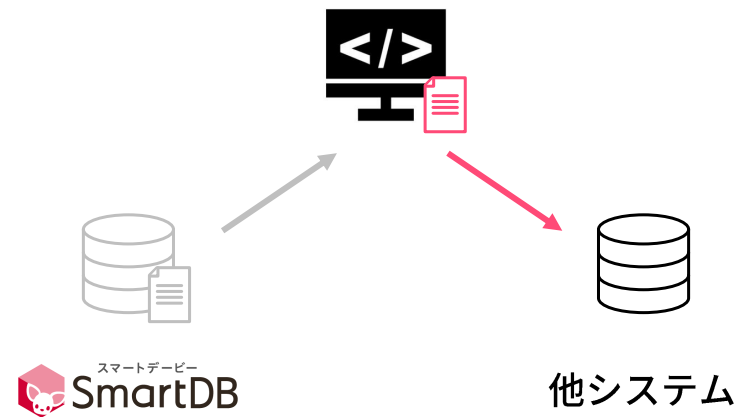
STEP 1



文書のデータを取得

- 文書詳細情報取得
- 指定したビューの文書一覧取得
- 文書一覧のCSV出力

STEP 2



他システムにデータを登録/更新/削除

他システムの提供するAPI

#	API	概要	利用シーン
1	文書詳細情報取得	指定された文書番号の文書データをJSONで取得 ※配置されたすべての部品データが取得されます	<ul style="list-style-type: none">1回の処理で1つの文書データを処理SmartDBでの処理発生時に実行 例) 申請書を連携
2	指定したビューの文書一覧取得	指定されたビューで表示される文書一覧をJSONで取得 ※指定したビューIDの表示項目にある部品データが取得されます	<ul style="list-style-type: none">1回の処理で複数の文書データを処理数十分、数時間おきに実行 例) 承認済みの申請書をまとめて連携
3	文書一覧のCSV出力	指定されたビューで表示される文書一覧をCSVで取得 ※指定したビューIDの表示項目にある部品データが取得されます	<ul style="list-style-type: none">1回の処理で大量の文書データを処理日時、週次で実行 例) マスタを連携

※業務要件などにより推奨されるAPIは異なる場合があります、実際にご利用する際は業務要件に適するAPIをご確認ください。

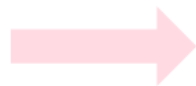
PINDB

ソフト型部品の行新規登録

リッチコメントの削除

SmartDBの文書を他システムに連携

他システムのデータをSmartDBの文書に連携



スマートデービー SmartDB

他システム



スマートデービー SmartDB

他システム

行更新

行削除

得

歴リスト

歴取得

の文書一

の文書計

出力

)

)

CSV入力

リスト取

の投稿

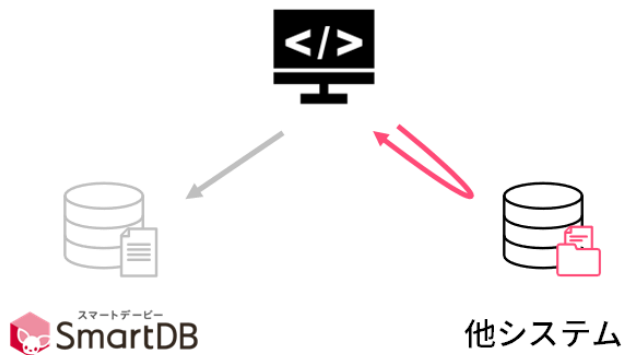
の詳細情

リッチコメントの更新

リスト型部品削除

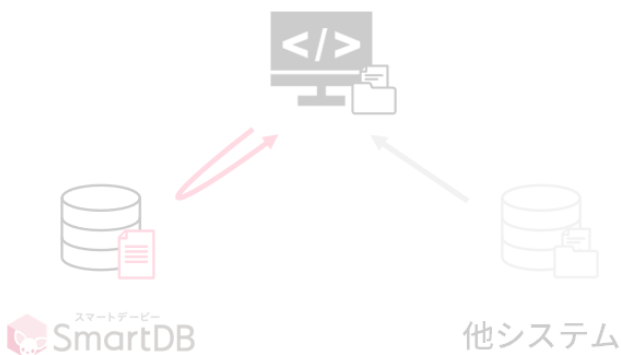
他システムのデータをSmartDBの文書に連携（文書新規登録）

STEP 1



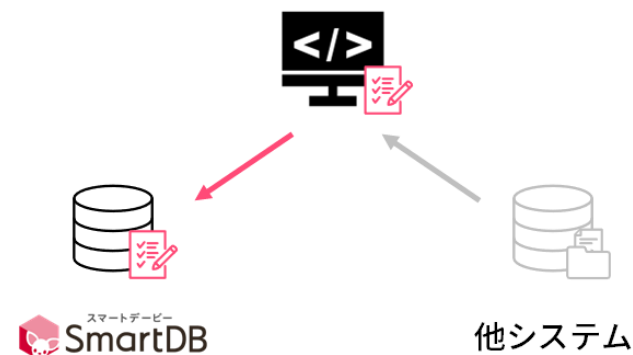
他システムのデータを取得
他システムの提供するAPI

STEP 2



対象文書の文書番号を取得
指定したビューの文書一覧取得

STEP 3

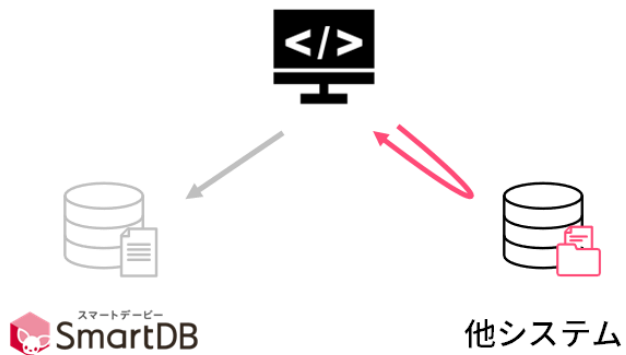


文書の新規登録/更新/削除
文書新規登録
文書一括操作
CSV入力（新規）

※STEP 3が「文書新規登録」の場合は不要

他システムのデータをSmartDBの文書に連携

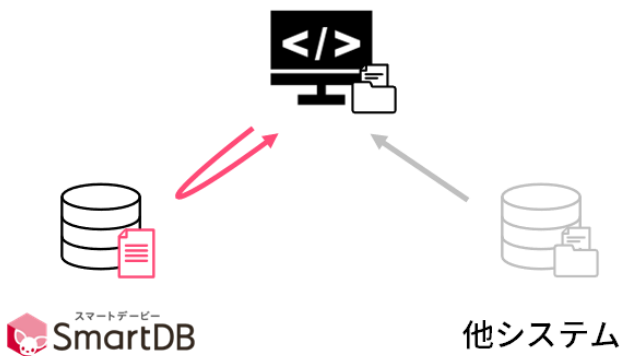
STEP 1



他システムのデータを取得

他システムの提供するAPI

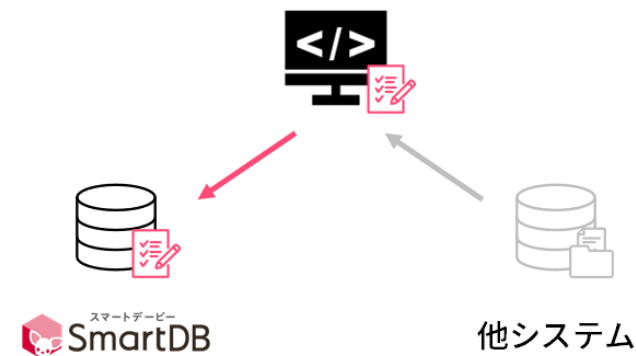
STEP 2



対象文書の文書番号を取得

指定したビューの文書一覧取得

STEP 3



文書の新規登録/更新/削除

文書更新 / 文書削除
文書一括操作
CSV入力 (更新)

STEP 2 「**文書番号を取得**」が必要な理由

元データ（外部システム）

項目	値
一意 法人番号	9011001040835
法人名	株式会社ドリーム・アーツ
担当者	○山×太
	⋮

SmartDBの文書データ

項目	値
文書番号	6
一意 法人番号	9011001040835
法人名	株式会社ドリーム・アーツ
担当者	○川×美
	⋮

1. REST API「文書更新」で変更対象の文書を指定するために、**文書番号**を利用している
文書更新：<https://{SmartDBのドメイン}.smartdb.jp/hibiki/rest/3/binders/{バインダID}/documents/{文書番号}>
2. 外部システムでは**文書番号**をデータとして持たない（ケースが多い）
3. 複数のシステムで**一意の項目（法人番号、社員ID）**を用いて、文書番号を取得する

#	API	概要	利用シーン
1	文書新規登録	指定された値で文書を登録 ※値を指定していない部品は設定された初期値で登録されます	<ul style="list-style-type: none"> 1回の処理で1つの文書データを処理 SmartDBでの処理発生時に実行 例) データを文書に連携
	文書更新	指定された文書番号の文書を更新 ※値を指定していない部品の値は更新されません	
	文書削除	指定された文書番号の文書を削除	
2	文書一括操作	複数文書の登録/更新/削除をまとめて実行	<ul style="list-style-type: none"> 1回の処理で複数の文書データを処理 数十分、数時間おきに実行 例) 確定データをまとめて文書に連携 ※バインダが異なる複数の文書に処理を行うことも可能
3	CSV入力（新規）	CSVで文書を新規登録	<ul style="list-style-type: none"> 1回の処理で大量の文書データを処理 日時、週次で実行 例) マスタを連携 ※対象のバインダは1つのみ
	CSV入力（更新）	CSVで文書を更新	

※業務要件などにより推奨されるAPIは異なる場合があります、実際にご利用する際は業務要件に適するAPIをご確認ください。

問題

よく使うバインダ、文書のREST APIの説明として正しいのは？

※ DCS / オンプレミスの場合、手順が異なる場合がございます。

1. 文書の登録、更新、削除を行う場合、一度に扱う文書数に応じてREST APIを使い分ける。
2. SmartDBの文書データを他システムに連携することはできない。
3. 他システムのデータをSmartDBのバインダ及び文書に連携することはできない。
4. 更新・削除する文書は文書番号以外でも指定できる。

問題

よく使うバインダ、文書のREST APIの説明として正しいのは？

※ DCS / オンプレミスの場合、手順が異なる場合がございます。

1. 文書の登録、更新、削除を行う場合、一度に扱う文書数に応じてREST APIを使い分ける。
2. SmartDBの文書データを他システムに連携することはできない。
3. 他システムのデータをSmartDBのバインダ及び文書に連携することはできない。

解説

1. 文書の登録、更新、削除のREST APIは複数あり、文書数などに応じて使い分ける
2. SmartDB REST APIを使えば、SmartDBと他システムを双方向に連携することができる
3. 文書の更新、削除を行う場合、文書番号によって対象文書を指定する必要がある。

1. 前回の復習
2. SmartDBのAPIの始め方
3. よく使うSmartDB REST API
 - バインダ / 文書
 - 業務プロセス
4. ハンズオン
5. まとめ
6. 次回予告

PROCESS

開始可能業務一覧取得
よく使う業務一覧取得
よく使う業務一覧に追加
よく使う業務一覧から削除
業務プロセス開始
開始済み一覧取得
実施済一覧取得
履歴一覧取得
ワークリスト取得
実施履歴取得
アクティビティ詳細情報取得
アクティビティ実施
文書情報に基づいて実行中アクティビティ情報取得
指定ユーザのすべての代行設定取得
代行設定新規作成
代行設定取得

代行設定更新
代行設定削除

PROCESS

開始可能業務一覧取得

よく使う業務一覧取得

よく使う業務一覧に追加

よく使う業務一覧から削除

業務プロセス開始

開始済み一覧取得

実施済一覧取得

履歴一覧取得

ワークリスト取得

実施履歴取得

アクティビティ詳細情報取得

アクティビティ実施

文書情報に基づいて実行中アクティビティ情報取得

指定ユーザのすべての代行設定取得

代行設定新規作成

代行設定取得

代行設定更新

代行設定削除

PROCESS

他システムの処理をトリガーに・・・

SmartDBで新規文書のWFを実行

SmartDBで既存文書のWFを実行



他システム

〇〇という処理を実行

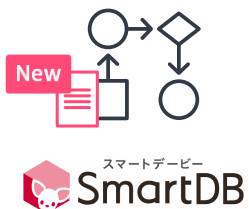
例) 人事マスタデータを更新



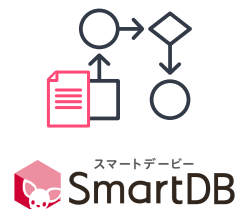
他システム

〇〇という処理を実行

例) 人事マスタデータを更新



業務プロセスを開始



業務プロセスを開始
アクティビティを実施

代行設定取得

PROCESS

他システムの処理をトリガーに・・・

SmartDBで新規文書のWFを実行

SmartDBで既存文書のWFを実行



他システム

〇〇という処理を実行

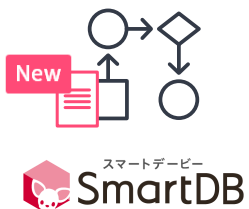
例) 人事マスタデータを更新



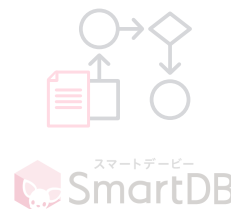
他システム

〇〇という処理を実行

例) 人事マスタデータを更新



業務プロセスを開始

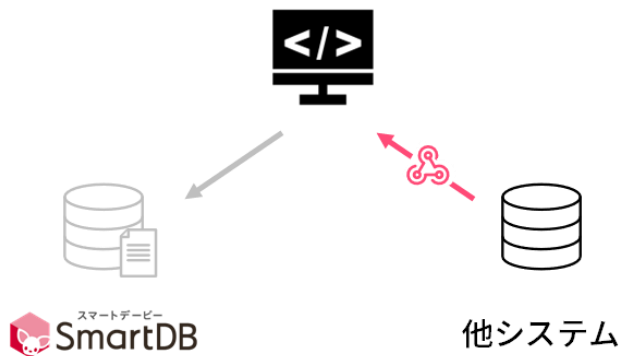


業務プロセスを開始
アクティビティを実施

代行設定取得

他システムの処理をトリガーにSmartDBで新規文書のWFを実行

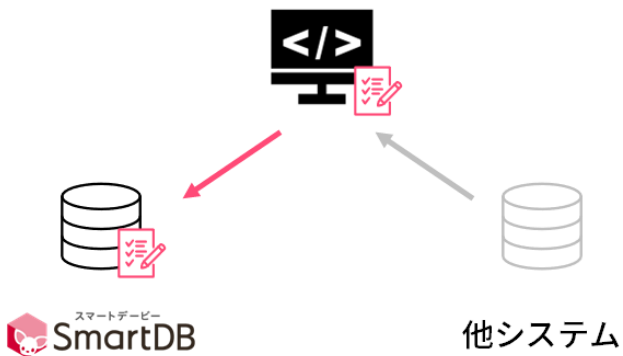
STEP 1



APIの実行プログラムを起動

他システムの提供するWebhook

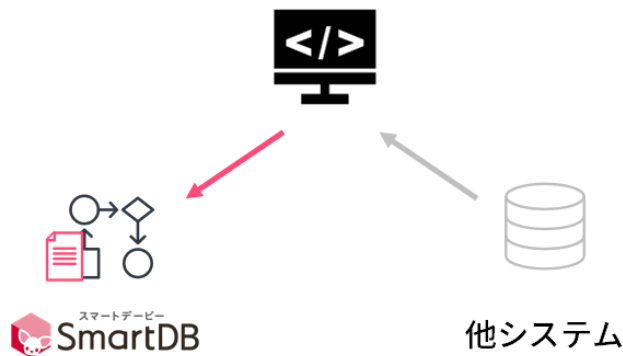
STEP 2



文書の新規登録

文書新規登録

STEP 3



登録した文書のWFを開始

業務プロセス開始

※STEP 2のレスポンスを利用して文書番号を指定

PROCESS

他システムの処理をトリガーに・・・

SmartDBで文書新規登録のWFを実行

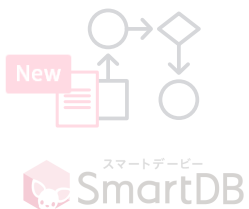
SmartDBで既存文書のWFを実行



他システム

〇〇という処理を実行

例) 人事マスタデータを更新



業務プロセスを開始

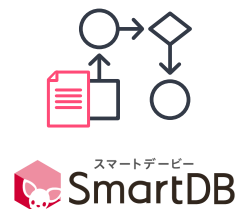
代行設定取得



他システム

〇〇という処理を実行

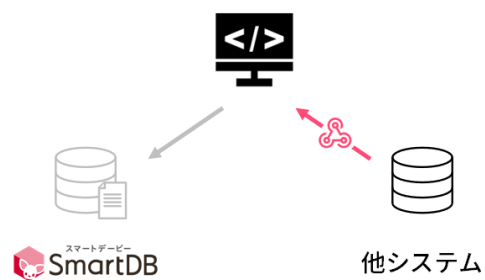
例) 人事マスタデータを更新



業務プロセスを開始
アクティビティを実施

他システムの処理をトリガーにSmartDBで既存文書のWFを実行

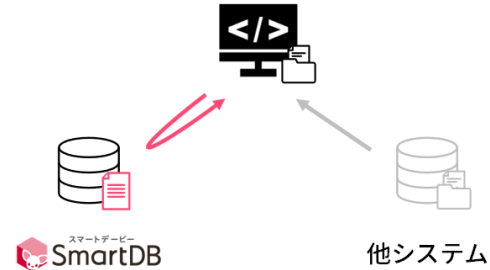
STEP 1



APIの実行プログラムを起動

他システムのWebhook

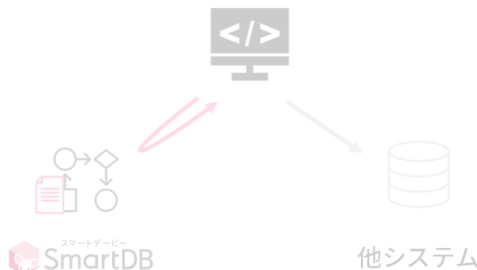
STEP 2



WFを開始する文書の
文書番号を取得

指定したビューの文書一覧取得

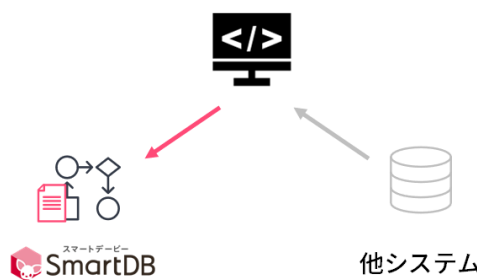
STEP 3



実施するアクティビティの
IDを取得

文書情報に基づいて実行中
アクティビティ情報取得

STEP 4



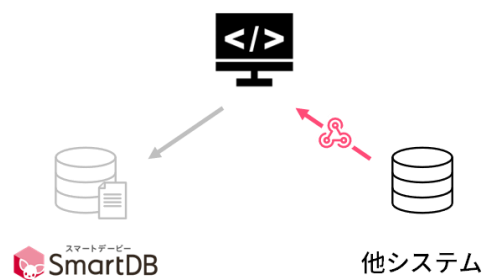
既存文書のWFを開始

業務プロセス開始

※STEP 4が「業務プロセス開始」の場合は不要

他システムの処理をトリガーにSmartDBで既存文書のWFを実行

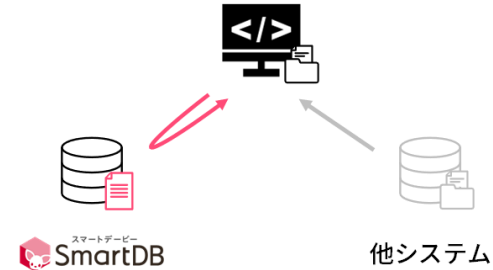
STEP 1



APIの実行プログラムを起動

他システムのWebhook

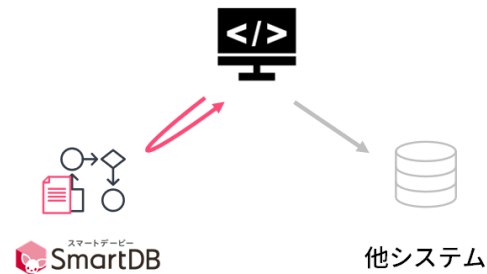
STEP 2



WFを開始する文書の
文書番号を取得

指定したビューの文書一覧取得

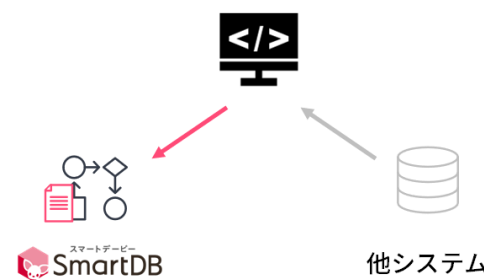
STEP 3



実施するアクティビティの
IDを取得

文書情報に基づいて実行中
アクティビティ情報取得

STEP 4



既存文書のWFの
アクティビティを実施

アクティビティ実施

問題

よく使う業務プロセスのREST APIの説明として正しいのは？

※ DCS / オンプレミスの場合、手順が異なる場合がございます。

1. 業務プロセス開始のREST APIでは、アクティビティIDの指定が必要となる。
2. アクティビティ実施のREST APIでは、アクティビティIDの指定が必要となる。
3. 他システムの処理を起点に、業務プロセスを開始することはできない。
4. 他システムの処理を起点に、アクティビティを実施することはできない。

問題

よく使う業務プロセスのREST APIの説明として正しいのは？

※ DCS / オンプレミスの場合、手順が異なる場合がございます。

1. 業務プロセス開始のREST APIでは、アクティビティIDの指定が必要となる。
2. **アクティビティ実施のREST APIでは、アクティビティIDの指定が必要となる。**
3. 他システムの処理を起点に、業務プロセスを開始することはできない。

解説

1. 「業務プロセス開始」では、基本的にプロセス定義番号、開始バインダID（キー）、開始文書の文書IDによって処理する業務プロセスを指定する。
2. 「アクティビティ実施」では、アクティビティIDによって処理するアクティビティを指定する。

1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

参照情報

リクエスト

ACCOUNT >

ROLE >

BINDER >

DOCUMENT >

FILE >

CATEGORY >

PROCESS >

LUXOR >

SESSION >

SmartDB REST API (v3)

Download OpenAPI specification: [Download](#)

SmartDB では HTTP アクセスで実行可能な REST API を利用することが可能です。

APIの実行形式

SmartDBのREST APIは、下記の形式で実行できます。

```
https://(SmartDBドメイン)/hibiki/rest/ (対象API)
```

REST API実行時のセッションについて

REST APIの実行にはSmartDBのセッションが必要となります。
セッションの利用方法は下記の3パターンが考えられます。

トークンを利用してREST APIを実行します。
※セッションの有効期限は、トークンによって異なります。

名前
00 スマラジ！ REST API基本編

有効期限
2025/1/4

バイндаトークン
FK8m_a65m0pFcXJlGCIaI0kAiBrv2v8p

トークンID
5018fb5c-6aa8-4bab-8cfe-2a2ff827b4

トークンタイプ
バイндаトークン

メモ
ユーザー向けイベント「スマラジ！ REST API基本編」で利用予定

1 文書一覧・ビュー情報

バイнда名	現在参照している一覧ビュー
通知用 (フレームあり)	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list
フレームなし	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true
ポートレット用	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true&portlet=true&parts.visible(title)=true&parts.visible(navi)=true&parts.visible(action)=true&parts.visible(criteria)=true&parts.visible(tree)=true&parts.visible(info)=true&parts.visible(cliboard)=true
通知用 (フレームあり)	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?viewId=10001
フレームなし	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true&viewId=10001
ポートレット用	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true&portlet=true&viewId=10001&parts.visible(title)=true&parts.visible(navi)=true&parts.visible(action)=true&parts.visible(criteria)=true&parts.visible(tree)=true&parts.visible(info)=true&parts.visible(cliboard)=true

優先順	項目	並び替え方向
1	[Number] 東議No.	降順 ▼



ハンズオンで使用するPowerShellスクリプト

※コミュニティサイトへの登録が必要です。

参照情報

リクエスト

どのREST APIを使えばいいかわかったけど・・・。
 どうやってリクエストを作ればいいの・・・？

ACCOUNT	>
ROLE	>
BINDER	>
DOCUMENT	>
FILE	>
CATEGORY	>
PROCESS	>
LUXOR	>
SESSION	>

SmartDB RE
 Download OpenAPI specifi
 SmartDBではHTTPアクセ
 APIの実行形式
 SmartDBのREST APIは、T
[https://\(SmartDBド](https://(SmartDBド)
 REST API実行時のセッ
 REST APIの実行にはSmart
 セッションの利用方法は下

リクエスト
 ヘッダ

リクエストヘッダ

リクエストボディ

名前
00 スマラジ！ REST API基本編

有効期限
2025/1/4

バインダトークン
FK8m_a65m0pF

トークンID
5018fb5c-6aa8-4bab-127b4

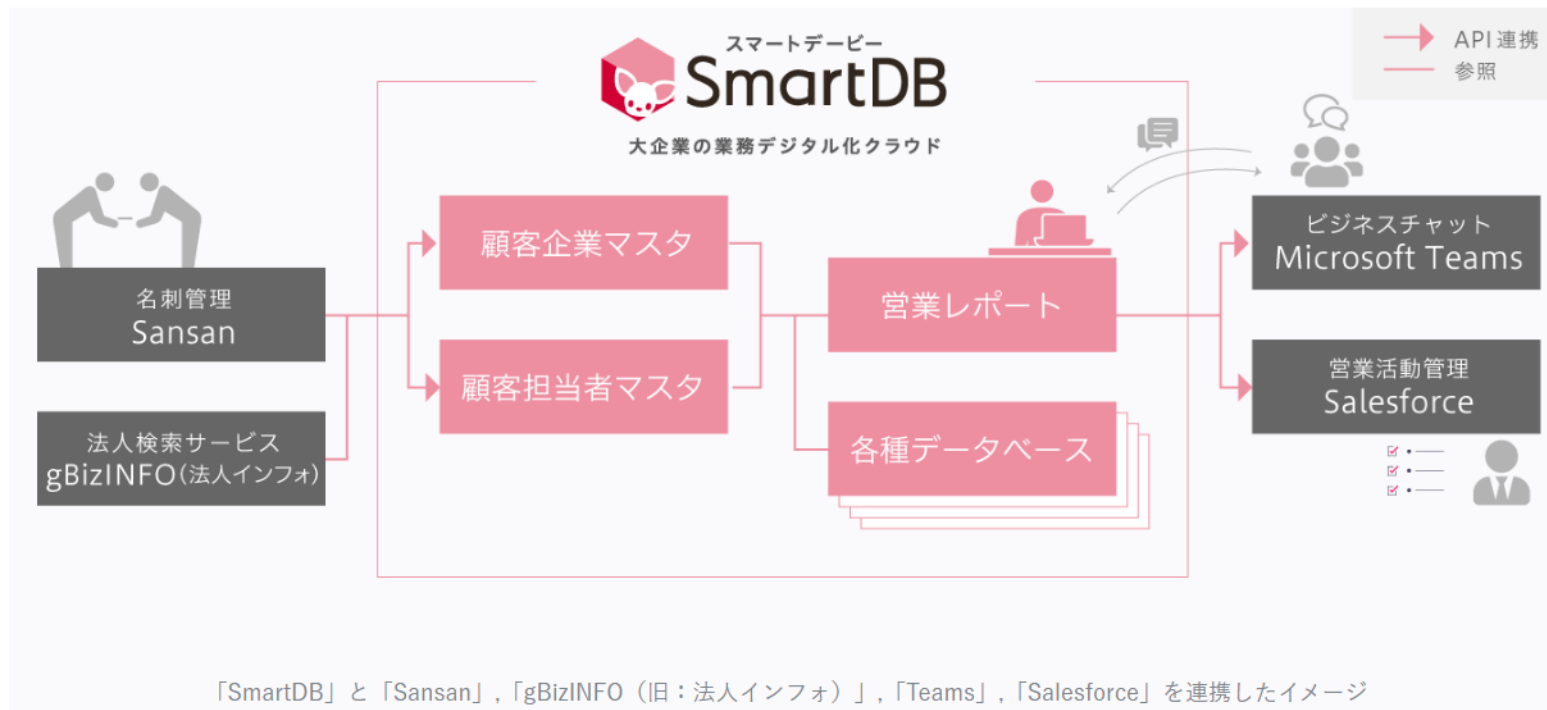
トークンタイプ
バインダト

メモ
ユーザー

1 優先順 項目 並び替え方向

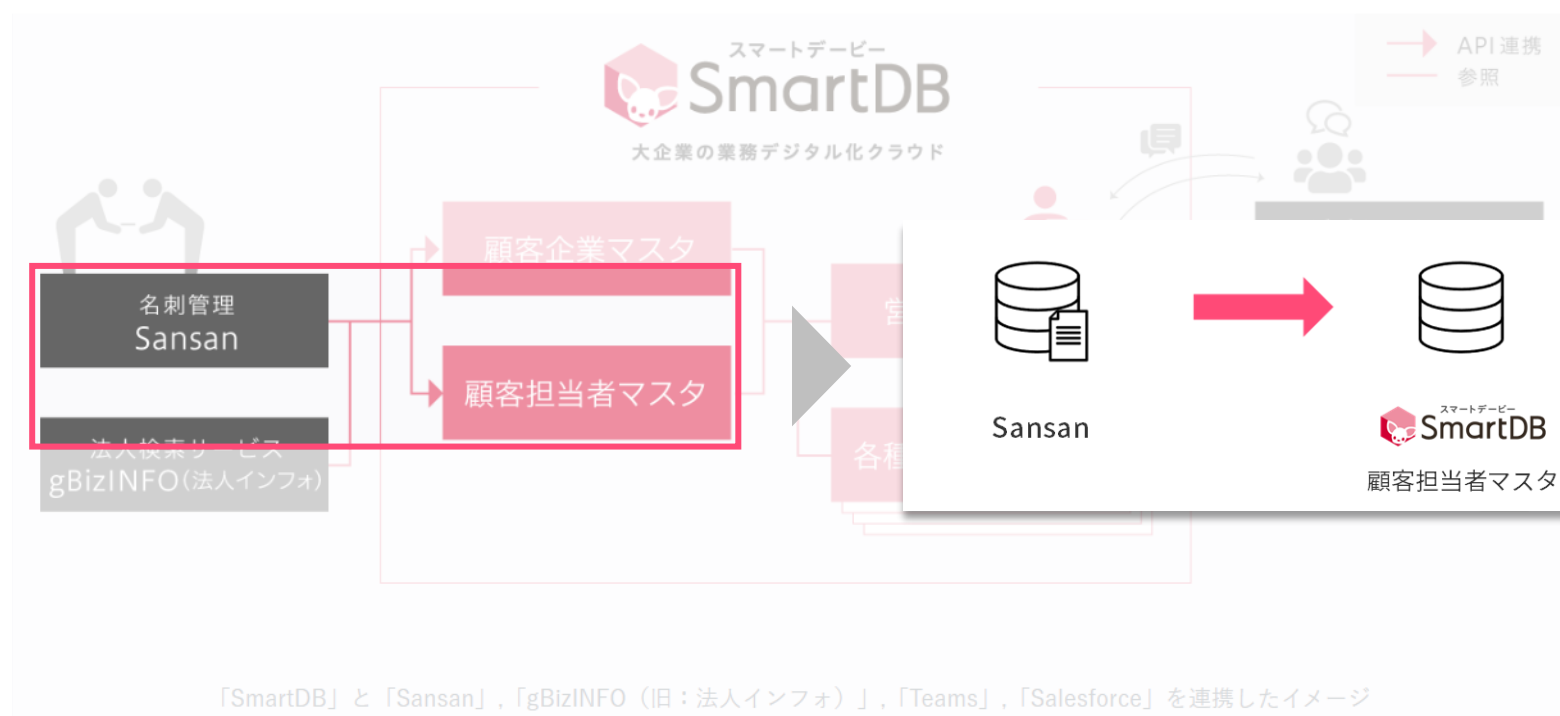
通知用 (フレームあり)	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list
フレームなし	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true
ポートレット用	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true&portlet=true&parts.visible(title)=true&parts.visible(nav)=true&parts.visible(action)=true&parts.visible(criteria)=true&parts.visible(tree)=true&parts.visible(info)=true&parts.visible(clipboard)=true
通知用 (フレームあり)	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?viewId=10001
フレームなし	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true&viewId=10001
ポートレット用	https://seminar.smartdb.jp/hibiki/smartdb/binder/13245/document/list?noMenu=true&portlet=true&viewId=10001&parts.visible(title)=true&parts.visible(nav)=true&parts.visible(action)=true&parts.visible(criteria)=true&parts.visible(tree)=true&parts.visible(info)=true&parts.visible(clipboard)=true

他システムの名刺情報からSmartDBの顧客担当者マスタを更新する



(出典) [API連携事例 | 「Salesforce」 × 「Sansan」 × 「SmartDB」 で営業の情報管理効率化 | SmartDB®【大企業の業務デジタル化クラウド】 \(dreamarts.co.jp\)](#)

他システムの名刺情報からSmartDBの顧客担当者マスタを更新する



(出典) [API連携事例 | 「Salesforce」 × 「Sansan」 × 「SmartDB」 で営業の情報管理効率化 | SmartDB®【大企業の業務デジタル化クラウド】 \(dreamarts.co.jp\)](#)

他システムの名刺情報からSmartDBの顧客担当者マスタを更新する

STEP 1



本講座では省略

Sansan Open APIを利用して
「人物ID」を取得

STEP 2



SmartDB REST APIと「人物ID」
を利用して文書番号を取得

※「指定したビューの文書一覧取得」を利用

STEP 3

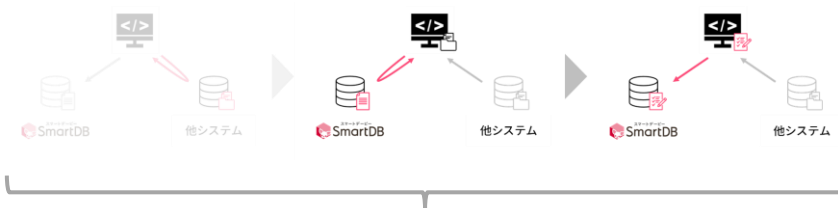


SmartDB REST APIと「文書番号」
を利用して文書のデータを更新

※「文書更新」を利用

通常の場合

1つのプログラム(コード)で一連の全てSTEPを実行する



```

# 1. サンプルデータ
$SampleData = @{
    "key" = "000880"
    "Title" = "テスト申請 (変更)"
}

# 2. 文書詳細取得 (GET)
$URI = "https://[SmartDB環境ドメイン].smartdb.jp"
+ "/hibiki/rest/3/binders/[バインダID]/views/[ビューID]/documents"
+ "?DocNumber=" + $SampleData.key + ":EXACT"
$Method = "GET"
$headers = @{
    "Authorization" = "Bearer [バインダトークン]"
}

Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $Body -UseBasicParsing -OutFile .\response_DocList.json

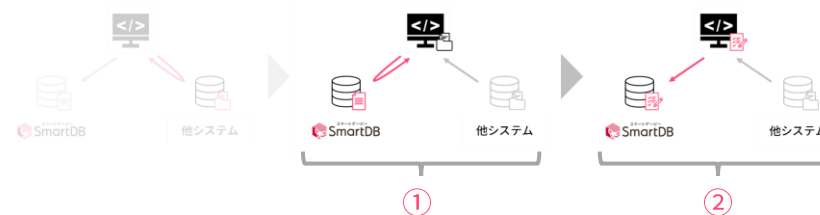
# 3. 2で取得した文書データから文書番号を抽出
$response = ((Get-Content .\response_DocList.json -Encoding UTF8) | ConvertFrom-Json)
$DocID = $response.documents[0].id

# 4. 文書更新 (PUT)
$URI = "https://[SmartDB環境ドメイン].smartdb.jp"
+ "/hibiki/rest/3/binders/10608/documents/"
+ $DocID
$Method = "PUT"
$headers = @{
    "Authorization" = "Bearer S2Y17UkU0sZ0IT17IK6nfh5c0s1s1n_oh"
}
$body = @{
    "DocTitle" = $SampleData.Title
}
$body = $body | ConvertTo-Json
$body = [System.Text.Encoding]::UTF8.GetBytes($body)

Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $body -UseBasicParsing -OutFile .\response_PUTDoc.json
    
```

本講座の場合

便宜上、STEPごとにプログラム(コード)を分割する



①

```

$URI = "[①プロトコル + ホスト名]"
+ "[②リソースパス]"
+ "?Title=[③件名]:EXACT"
$Method = "[④HTTPメソッド]"
$headers = @{
    "[⑤認証用のヘッダ]" = "Bearer [⑥バインダトークン]"
}

Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $Body -UseBasicParsing -OutFile .\response_DocList.json
    
```

+

②

```

$URI = "[①プロトコル + ホスト名]"
+ "[②リソースパス]"
$Method = "[③HTTPメソッド]"
$headers = @{
    "[④認証用のヘッダ]" = "Bearer [⑤バインダトークン]"
}
$body = @{
    "Budget" = "[⑥予算額]"
    "Reason" = "[⑦起案理由]"
    "Contents" = "[⑧起案内容]"
}

Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $body -UseBasicParsing -OutFile .\response_PUTDoc.json
    
```

変更前

一覧 編集 再利用 更新履歴 削除

🔍 ブックマーク

文書タイトル	夢野芸太郎		
文書番号	1	更新	10:35 👤 中川 嵩之

顧客マスタ

■ 登録情報

登録日	2024/02/29
人物ID	AA0001
氏名	夢野芸太郎
氏名 (フリガナ)	ユメノゲイタロウ
会社名	株式会社ドリーム・アーツ
部署名	営業部

■ 担当営業者名

担当営業者	👤 中川 嵩之
-------	---------

🔍 ブックマーク

一覧 編集 再利用 更新履歴 削除

変更後

一覧 編集 再利用 更新履歴 削除

🔍 ブックマーク

文書タイトル	夢野芸太郎		
文書番号	1	更新	10:36 👤 中川 嵩之

顧客マスタ

■ 登録情報

登録日	2024/02/29
人物ID	AA0001
氏名	夢野芸太郎
氏名 (フリガナ)	ユメノゲイタロウ
会社名	株式会社ドリーム・アーツ
部署名	経営企画部

■ 担当営業者名

担当営業者	👤 中川 嵩之
-------	---------

🔍 ブックマーク

部署異動に伴う名刺の更新情報を顧客担当者マスタに反映

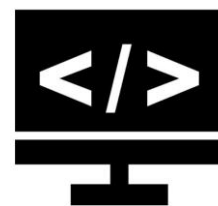
PowerShellでREST APIを実行する方法を確認

1. URI
 2. HTTPメソッド
 3. リクエストボディ
 4. リクエストヘッダ
- の指定

+

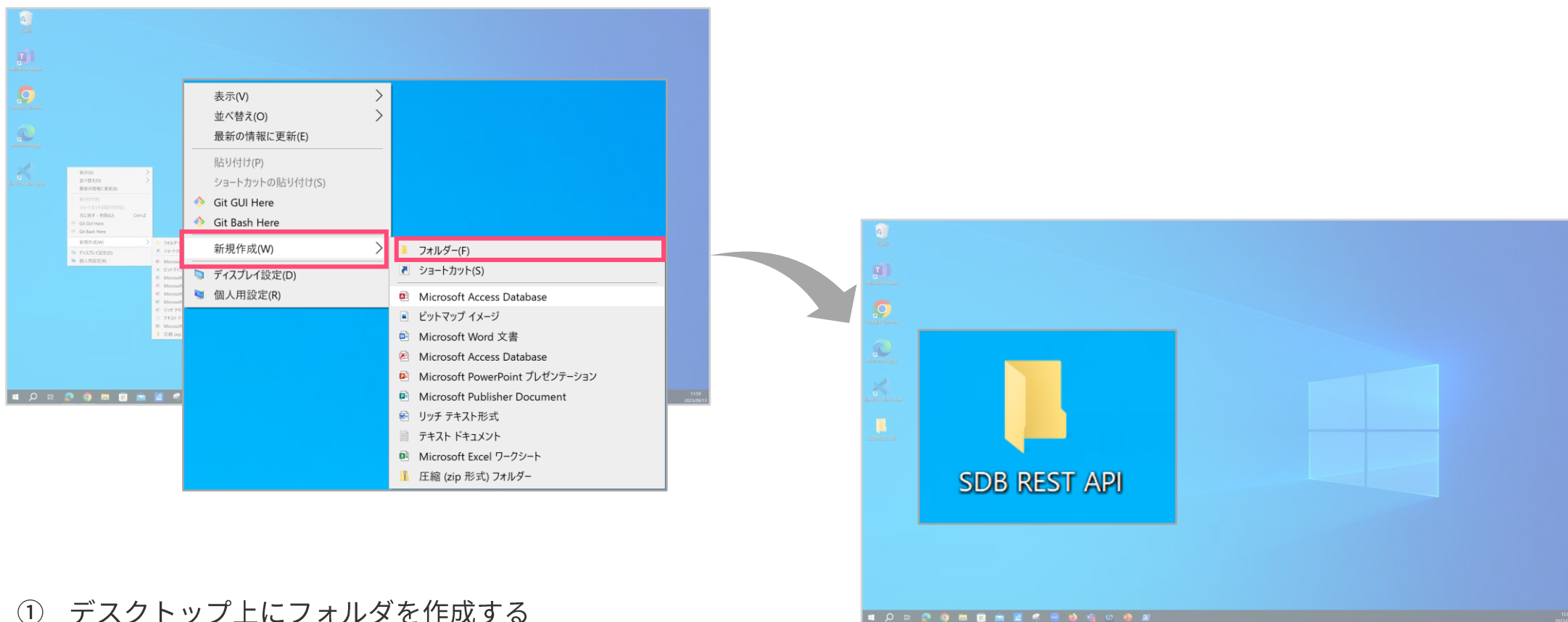
Invoke-WebRequest : REST APIを実行するコマンド

スマートデータベース
SmartDB



PowerShell

デスクトップにハンズオン用のフォルダを作成



- ① デスクトップ上にフォルダを作成する
- ② フォルダに「SDB REST API」と命名する

他システムの名刺情報からSmartDBの顧客担当者マスタを更新する

STEP 1



本講座では省略

Sansan Open APIを利用して
「人物ID」を取得

STEP 2



SmartDB REST APIと「人物ID」
を利用して文書番号を取得

※「指定したビューの文書一覧取得」を利用

STEP 3



SmartDB REST APIと「文書番号」
を利用して文書のデータを更新

※「文書更新」を利用

URIのプロトコル、リソースパスを指定

RestDocV3

指定したビューの文書一覧取得

指定されたバインダ、およびビューで表示される文書一覧を返却します。
返却される文書の部品データはURLに指定したビューIDの表示項目にある部品のみです。
表示項目に指定されていない部品のデータは返却されません。

AUTHORIZATIONS: > (cookieAuth and csrfToken)

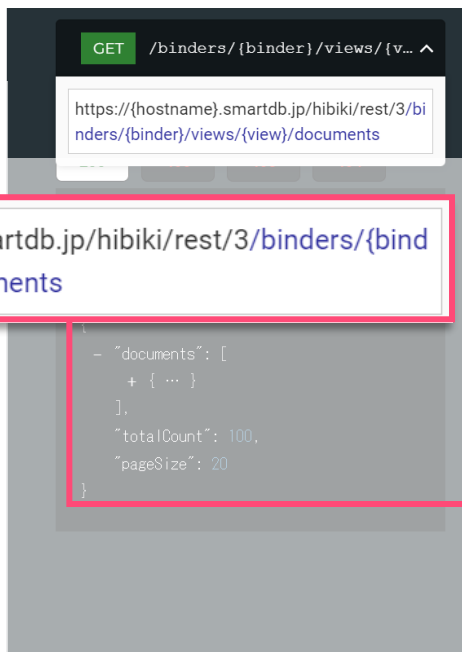
PATH PARAMETERS

binder > required	整数値 (integer) or 文字列 (string) バインダIDもしくはキー
view > required	integer or string ビューIDもしくはキー

QUERY PARAMETERS

pageNumber	integer <int32> Default: 1 ページ番号
pageSize	integer <int32>

`https://{hostname}.smartdb.jp/hibiki/rest/3/binders/{binder}/views/{view}/documents`



リクエスト

\$URI = “[①プロトコル + ホスト名 + リソースパス]”`
+ “?personalId=[②人物ID]:EXACT”

\$Method = “[③HTTPメソッド]”

\$Headers = @{

“[④認証用のヘッダ]” =

"Content-type"="applic

}

\$URI = “[①プロトコル + ホスト名 + リソースパス]”`
+ “?personalId=[②人物ID]:EXACT”

Invoke-WebRequest \$URI -Method \$Method -Headers \$Headers -Body \$Body -UseBasicParsing -OutFile .%response_DocList.json

\$URI = “`https://{hostname}.smartdb.jp/hibiki/rest/3/binders/{binder}/views/{view}/documents`”`
+ “?personalId=[②人物ID]:EXACT”

URIのホスト名、リソースパラメータを指定

SmartDBの文書一覧情報

リクエスト

通知用 (フレームあり) <https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?viewId=10004>

通知用 (フレームあり)	https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?viewId=10004
一覧ビューのURL	フレームなし https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?noMenu=true&viewId=10004
	ポートレット用 https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?noMenu=true&portlet=true&viewId=10004&parts.visible(title)=true&parts.visible(nav)=true&parts.visible(action)=true&parts.visible(criteria)=true&parts.visible(tree)=true&parts.visible(info)=true&parts.visible(cookieboard)=true

\$URI = “[①プロトコル + ホスト名 + リソースパス]”`
 + “?personalId=[②人物ID]:EXACT”

\$Method = “[③HTTPメソッド]”

\$URI = “https://{hostname}.smartdb.jp/hibiki/rest/3/binders/{binder}/views/{view}/documents”`
 + “?personalId=[②人物ID]:EXACT”

Invoke-WebRequest \$URI -Method \$Method -Headers \$Headers -Body \$Body -UseBasicParsing -OutFile .%response_DocList.json

\$URI = “https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/documents”`
 + “?personalId=[②人物ID]:EXACT”

URIのクエリパラメータを指定

他システムから取得した元データ

リクエスト

変更後

項目	値
人物ID	AA0001
氏名	夢野 芸
会社名	株式会社ドリーム・アーツ
部署名	経営企画部
	⋮

```
$URI = “[①プロトコル+ホスト名+リソースパス]” `
+ “?personalId=[②人物ID]:EXACT”
```

```
$Method = “[③HTTPメソッド]”
```

```
$Headers = @{
```

```
$URI = “https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/documents” `
+ “?personalId=[②人物ID]:EXACT”
```

```
Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .\response_DocList.json
```



```
$URI = “https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/document” `
+ “?personalId=AA0001:EXACT”
```



HTTPメソッドを指定

RestDocV3

リクエスト

指定したビューの文書一覧取得

指定されたバインダ、およびビューで表示される文書一覧を返却します。
返却される文書の部品データは、
です。
表示項目に指定されていない部

AUTHORIZATIONS: > (cookieAuth and csrfTokenAuth) or binderTokenAuth

PATH PARAMETERS

binder > required 整数値 (integer) or 文字列 (string)
バインダIDもしくはキー

view > required integer or string
ビューIDもしくはキー

QUERY PARAMETERS

pageNumber integer <int32>
Default: 1
ページ番号

pageSize integer <int32>

```
GET /binders/{binder}/views/{view}/documents
```

\$URI = “[①プロトコル+ホスト名+リソースパス]”`
+ “?personalId=[②人物ID]:EXACT”

\$Method = “[③HTTPメソッド]”

\$Headers = @{

“[④認証用のヘッダ]” = “Bearer [⑤バインダトークン]”

“Content-type” = “application/json”

\$Method = “[③HTTPメソッド]”

Invoke-WebRequest \$URI -Method \$Method -Headers \$Headers -Body \$Body -UseBasicParsing -OutFile .%response_DocList.json

\$Method = “GET”

HTTPヘッダの認証情報の方式を指定

RestDocV3

REST API実行時のセッションについて

REST APIの実行にはSmartDBのセッションが必要となります。
セッションの利用方法は下記の3パターンが考えられます。

- バインダトークンを利用して実行する。※バインダAPIオプション限定

管理画面の「バインダトークン」画面で発行したトークンを利用してREST APIを実行します。

実行時には後述の形式でHTTP Headerにトークンを指定してください。
バインダトークンを利用してAPIを実行する場合、下記の特徴を持ちます。

- トークンを実行するためのロボットユーザとして実行されます。
※ロボットユーザのタイムゾーンは+0900
- ParameterのcsrfTokenは不要です。
- バインダを指定するAPIの場合、トークンを設定したバインダにのみ権限

```
Authorization: Bearer {トークン}
```

HTTPヘッダーに以下を指定してください。

```
Authorization: Bearer {トークン}
```

リクエスト

```
$URI = "[①プロトコル+ホスト名+リソースパス]"`  
+ "?personalId=[②人物ID]:EXACT"
```

```
$Method = "[③HTTPメソッド]"
```

```
$Headers = @{  
    "[④認証用のヘッダ]" = "Bearer [⑤バインダトークン]"  
    "Content-type" = "application/json"  
}
```

```
Invoke-WebRequest $URI -Method $Method -Headers $Headers -ContentType application/json -ResponseFormat file..%response_DocList.json
```

```
"[④認証用のヘッダ]" = "Bearer [⑤バインダトークン]"  
"Content-type" = "application/json"
```

```
$Headers = @{  
    "Authorization" = "Bearer [⑤バインダトークン]"  
    "Content-type" = "application/json"  
}
```

認証情報のバイндаトークンを指定

SmartDBのバイндаトークン管理画面

リクエスト

名前
00 スマラジ！ REST API基本編

有効期限
2025/1/4

バイндаトークン
FK8m_a65m0pFcxJIGCiAi0kAiBrv2v8p

バイндаトークン
FK8m_a65m0pFcxJIGCiAi0kAiBrv2v8p

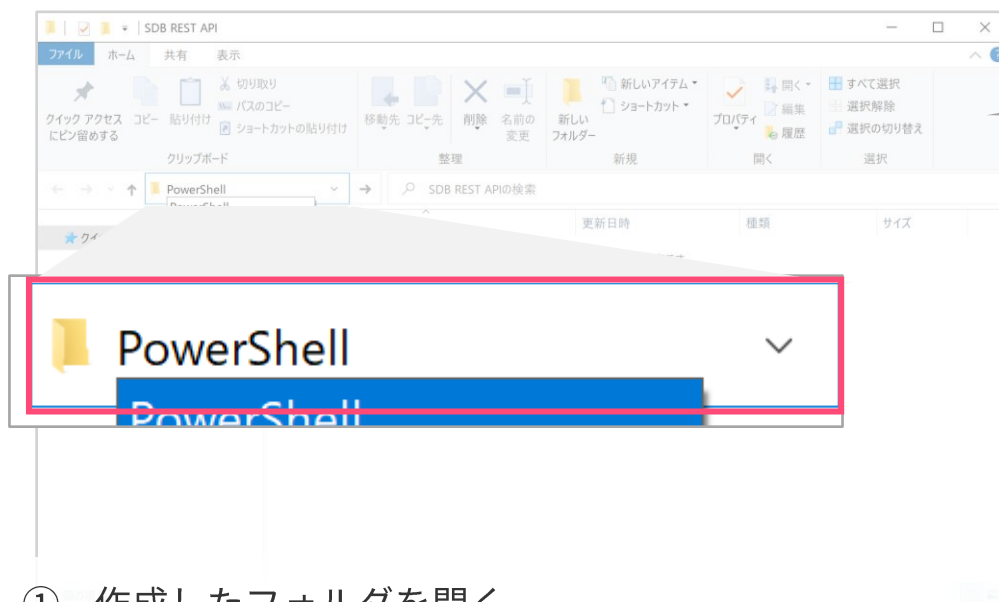
メモ
ユーザー向けイベント「スマラジ！ REST API基本編」で利用予定

```
$URI = "[①プロトコル+ホスト名+リソースパス]"`  
+ "?personalId=[②人物ID]:EXACT"  
$Method = "[③HTTPメソッド]"  
$Headers = @{  
    "[④認証用のヘッダ]" = "Bearer [⑤バイндаトークン]"  
    "Content-type" = "application/json"  
}  
Invoke-WebRequest $URI -Method $Method -Headers $Headers -ContentType application/json -File .\response_DocList.json
```

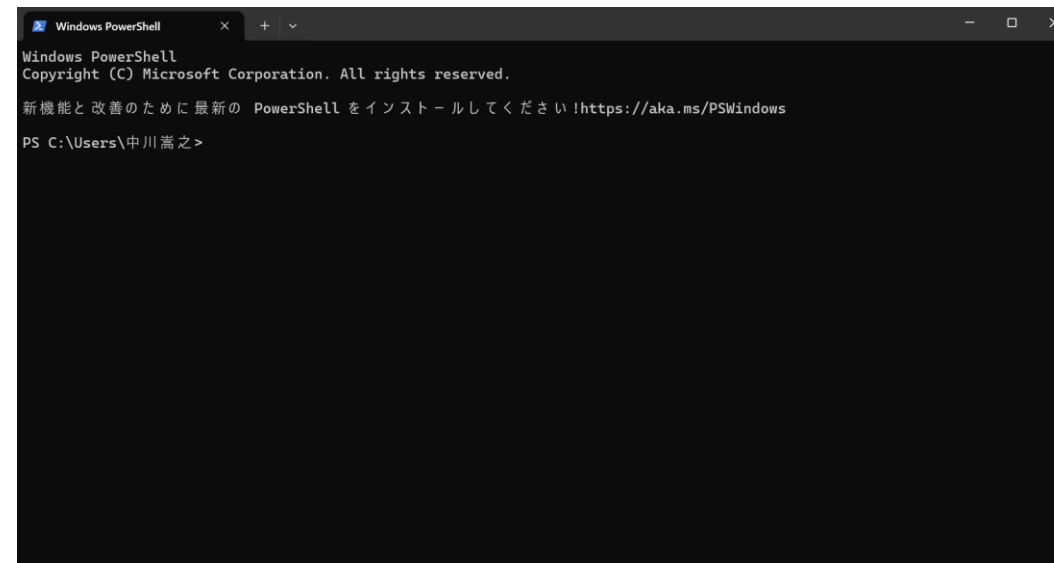
↓

```
$Headers = @{  
    "Authorization" = "Bearer FK8m_a65m0pFcxJIGCiAi0kAiBrv2v8p"  
    "Content-type" = "application/json"  
}
```

作成したフォルダからPowerShellを起動



- ① 作成したフォルダを開く
- ② アドレスバーに「PowerShell」と入力する
- ③ Enterキーを押す



※ログファイルの出力先を今回作成したフォルダにするため、フォルダから起動しています。

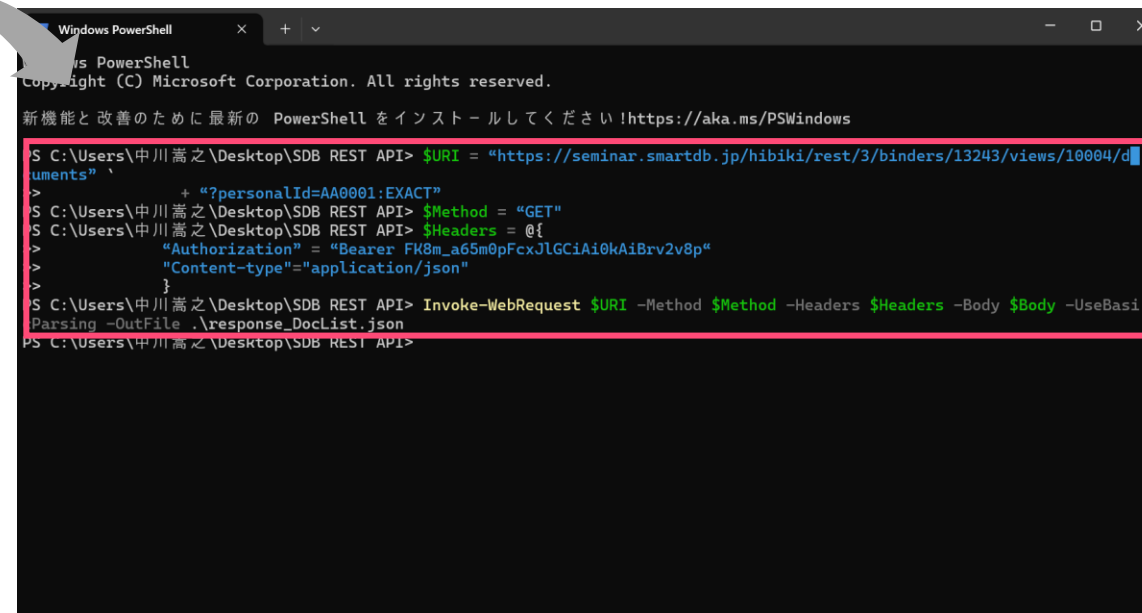
作成したコードをPowerShellで実行

【記入用】 文書番号を取得するコード

DreamArts

```
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/documents" `
+ "?personalId=AA0001:EXACT"
$Method = "GET"
$headers = @{
    "Authorization" = "Bearer FK8m_a65m0pFcXJlGCiAi0kAiBrv2v8p"
    "Content-type" = "application/json"
}
Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $Body -UseBasicParsing -OutFile .\response_DocList.json
```

- ① 作成したコードをコピーする
- ② PowerShellに貼り付ける
- ③ Enterキーで実行する



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

S C:\Users\中川嵩之\Desktop\SDB REST API> $URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/d
uments" `
> + "?personalId=AA0001:EXACT"
S C:\Users\中川嵩之\Desktop\SDB REST API> $Method = "GET"
S C:\Users\中川嵩之\Desktop\SDB REST API> $headers = @{
>     "Authorization" = "Bearer FK8m_a65m0pFcXJlGCiAi0kAiBrv2v8p"
>     "Content-type" = "application/json"
> }
S C:\Users\中川嵩之\Desktop\SDB REST API> Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $Body -UseBasi
Parsing -OutFile .\response_DocList.json
PS C:\Users\中川嵩之\Desktop\SDB REST API>
```

実行結果を確認

成功

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\中川嵩之\Desktop\SDB REST API> $URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/documents" `
>> + "?personalId=AA0001:EXACT"
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Method = "GET"
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Headers = @{
>> "Authorization" = "Bearer FK8m_a65m0pFcxJLGCiAi0kAiBrv2v8p"
>> "Content-type"="application/json"
>> }
PS C:\Users\中川嵩之\Desktop\SDB REST API> Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .\response_DocList.json
PS C:\Users\中川嵩之\Desktop\SDB REST API>
```

エラーが出力されない

失敗

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\中川嵩之> $URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/views/10004/documents" `
>> + "?personalId=AA0001:EXACT"
PS C:\Users\中川嵩之> $Method = "GET"
PS C:\Users\中川嵩之> $Headers = @{
>> "Authorization" = "Bearer FK8m_a65m0pFcxJLGCiAi0kAiBrv2v8p"
>> "Content-type"="application/json"
>> }
PS C:\Users\中川嵩之> Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .\response_DocList.json
Invoke-WebRequest : リモート サーバがエラーを返しました: (401) 許可されていません
発生場所 行:1 文字:1
+ Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body ...
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
+ FullyQualifiedErrorId : WebCmdLetWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

PS C:\Users\中川嵩之>
```

赤字でエラーが出力

他システムの名刺情報からSmartDBの顧客担当者マスタを更新する

STEP 1



本講座では省略

Sansan Open APIを利用して
「人物ID」を取得

STEP 2



SmartDB REST APIと「人物ID」
を利用して文書番号を取得

※「指定したビューの文書一覧取得」を利用

STEP 3



SmartDB REST APIと「文書番号」
を利用して文書のデータを更新

※「文書更新」を利用

URIのプロトコル、リソースパスを指定

RestDocV3

文書更新

指定された文書番号の文書を更新します。
引数に指定しなかった部品の値は変更されません。

AUTHORIZATIONS: > (cookieAuth and csrfTokenAuth) or binderTokenAuth

PATH PARAMETERS

binder > required
整数値 (integer) 或文字
バイナリIDもしくは:
document > required
integer <int64>
文書番号

QUERY PARAMETERS

skipItemControlCheck boolean
Default: false
部品制御の入力チェック (必須入力制御、入力値チェック、重複チェック) をスキップするか。
docType string
Default: "NORMAL"
Enum: "NORMAL" | "DRAFT"
文書タイプを指定する
• NORMAL: 正式文書
• DRAFT: 下書き文書
deleteTempFile boolean
Default: true
部品値に利用した一時ファイルを削除するか

https://{hostname}.smartdb.jp/hibiki/rest/3/binders/{binder}/documents/{document}

PUT /binders/{binder}/documents/{document}

Response samples

200 400 403 404

```
application/json
{
  "id": 1,
  "name": "string",
  "created": {
    "id": 1000028,
    "name": "サンプルユーザ",
    "status": 0,
    "recognizeInfo": "string",

```

リクエスト

\$URI = “[①プロトコル+ホスト名+リソースパス]”

\$Method = “[②HTTPメソッド]”

\$Headers = @{

“[③認証用のヘッダ]” = “Bearer [④バインダートークン]”

“Content-type” = “application/json”

}

\$Body = @{

“[⑤部署名の部品キー]” = “[⑥部署名の値]”

}

\$Body = \$Body | ConvertTo-Json

\$URI = “[①プロトコル+ホスト名+リソースパス]”

\$URI = ‘https://{hostname}.smartdb.jp/hibiki/rest/3/binders/{binder}/documents/{document}’

URIのホスト名、リソースパラメータを指定

SmartDBの文書一覧情報

リクエスト

通知用 (フレームあり) <https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?viewId=10004>

通知用 (フレームあり)	https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?viewId=10004
一覧ビューのURL	フレームなし https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?noMenu=true&viewId=10004
ポートレット用	https://seminar.smartdb.jp/hibiki/smartdb/binder/13243/document/list?noMenu=true&portlet=true&viewId=10004&parts.visible(title)=true&parts.visible(nav)=true&parts.visible(action)=true&parts.visible(criteria)=true&parts.visible(tree)=true&parts.visible(info)=true&parts.visible(coboard)=true

初期の並び順

優先順		昇順
1	[Frigana] フリガナ	

閉じる

```

$URI = “[①プロトコル+ホスト名+リソースパス]”
$Method = “[②HTTPメソッド]”
$headers = @{
    “[③認証用のヘッダ]” = “Bearer [④バインダトークン]”
}
$URI = “https://{hostname}.smartdb.jp/hibiki/rest/3/binders/{binder}/documents/{document}”

$body = @{
    “[⑤部署名の部品キー]” = “[⑥部署名の値]”
}
$body = $body | ConvertTo-Json
$body = [System.Text.Encoding]::UTF8.GetBytes($body)
$URI = “https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/documents/{document}”
    
```

Diagram illustrating the mapping of parameters from the notification URL to the request URI. Red boxes highlight 'seminar.smartdb.jp' and '13243' in the notification URL, and 'seminar.smartdb.jp' and '13243' in the request URI. Red arrows show the flow of information from the notification URL to the request URI.

URIのリソースパラメータを指定

前のSTEPで取得したデータ

リクエスト

```

101 {
102   "mustInput": false,
103   "decorator": {
104     "descriptionFontStyle": "color:#000000;",
105     "backgroundcolor": "#ffffff"
106   },
107   "displayDescriptionForEditingOnly": false,
108   "displayDescriptionInTooltip": false
109 },
110 {
111   "value": "部署部",
112   "reusable": true,
113   "name": "部署名",
114   "key": "departmentName",
115   "id": 10005,
116   "type": "Text",
117   "editable": true,
118   "mustInput": false,
119   "decorator": {
120     "descriptionFontStyle": "color:#000000;",
121     "backgroundcolor": "#ffffff"
122   },
123   "displayDescriptionForEditingOnly": false,
124   "displayDescriptionInTooltip": false
125 },
126 ],
127 "constraint": {
128   "counters": [],
129   "id": 1
130 },
131 },
132 "created": {
133   "key": "t_nakagawa@dreamarts.co.jp",
134   "name": "中川 浩志",
135   "primary": 2000000,
136   "primaryGroupName": "株式会社ナリーム・アーツ",
137   "email": "t_nakagawa@dreamarts.co.jp",
138   "disabled": false,
139   "status": 0,
140   "lang": {
141     "ja": {

```

\$URI = “[①プロトコル+ホスト名+リソースパス]”

\$Method = “[②HTTPメソッド]”

\$Headers = @{

“[③認証用のヘッダ]” = “Bearer [④バインダトークン]”

\$URI = “https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/documents/{document}”

\$Body = @{

“[⑤部署名の部品キー]” = “[⑥部署名の値]”

\$Body = \$Body | ConvertTo-Json

\$Body

Invoke-WebRequest -Uri \$URI -Method \$Method -Headers \$Headers -Body \$Body -ContentType json

HTTPメソッドを指定

RestDocV3

文書更新

指定された文書番号の文書を更新します。
引数に指定しなかった部品の値は変更されません。

AUTHORIZATIONS: > (cookie)

PATH PARAMETERS

PUT /binders/{binder}/documents/{document}

binders > required
整数値 (integer) または文字列 (string)
バインダIDもしくはキー

document > required
integer <int64>
文書番号

QUERY PARAMETERS

skipItemControlCheck boolean
Default: false
部品制御の入力チェック (必須入力制御、入力値チェック、重複チェック) をスキップするか。

docType string
Default: "NORMAL"
Enum: "NORMAL" | "DRAFT"
文書タイプを指定する

- NORMAL: 正式文書
- DRAFT: 下書き文書

deleteTempFile boolean
Default: true
部品値に利用した一時ファイルを削除するか

PUT /binders/{binder}/documents/{document}

```

{
  "property1": "string",
  "property2": "string"
}
    
```

Response samples

200 400 403 404

Content type
application/json

```

{
  "id": 1,
  "name": "string",
  "created": {
    "id": 1000028,
    "name": "サンプルユーザ",
    "status": 0,
    "recognizeInfo": "string",
    }
}
    
```

リクエスト

\$URI = "[①プロトコル+ホスト名+リソースパス]"

\$Method = "[②HTTPメソッド]"

\$Headers = @{

"[③認証用のヘッダ]" = "Bearer [④バインダトークン]"

"Content-type" = "application/

\$Method = "[②HTTPメソッド]"

\$Body = @{

"[⑤部署名の部品キー]" = "[⑥部署名の値]"

\$Body = \$Body | ConvertTo-Json

\$Body = [System.Text.Encoding]::UTF8.GetBytes

Invoke-WebRequest \$URI -Method \$Method -He

\$Method = "PUT"

BasicParsing -OutFile .¥response_PUTDoc.json

HTTPヘッダの認証情報の方式を指定

RestDocV3

REST API実行時のセッションについて

REST APIの実行にはSmartDBのセッションが必要となります。
セッションの利用方法は下記の3パターンが考えられます。

- バインダトークンを利用して実行する。※バインダAPIオプション限定

管理画面の「バインダトークン」画面で発行したトークンを利用してREST APIを実行します。

実行時には後述の形式でHTTP Headerにトークンを指定してください。
バインダトークンを利用してAPIを実行する場合、下記の特徴を持ちます。

- トークンを実行するためのロボットユーザとして実行されます。
※ロボットユーザのタイムゾーンは+0900
- ParameterのcsrfTokenは不要です。
- バインダを指定するAPIの場合、トークンを設定したバインダにのみ権限

```
Authorization: Bearer {トークン}
```

HTTPヘッダーに以下を指定してください。

```
Authorization: Bearer {トークン}
```

リクエスト

```
$URI = "[①プロトコル+ホスト名+リソースパス]"
```

```
$Method = "[②HTTPメソッド]"
```

```
$Headers = @{
```

```
    "[③認証用のヘッダ]" = "Bearer [④バインダトークン]"
```

```
    "Content-type" = "application/json"
```

```
}
```

```
$Body = @{
```

```
    "[⑤部署名の部品キー]" =
```

```
$Body = $Body | ConvertTo-Json
```

```
$Body = [System.Text.Encoding
```

```
Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .%response_PUTDoc.json
```

```
$Headers = @{
    "[④認証用のヘッダ]" = "Bearer [⑤バインダトークン]"
    "Content-type" = "application/json"
}
```

```
$Headers = @{
    "Authorization" = "Bearer [⑤バインダトークン]"
    "Content-type" = "application/json"
}
```

認証情報のバインダトークンを指定

SmartDBのバインダトークン管理画面

リクエスト

名前
00 スマラジ！ REST API基本編

有効期限
2025/1/4

バインダトークン
FK8m_a65m0pFcxJIGCiAi0kAiBrv2v8p

バインダトークン
FK8m_a65m0pFcxJIGCiAi0kAiBrv2v8p

メモ
ユーザー向けイベント「スマラジ！ REST API基本編」で利用予定

```
$URI = "[①プロトコル+ホスト名+リソースパス]"
$Method = "[②HTTPメソッド]"
$Headers = @{
    "[③認証用のヘッダ]" = "Bearer [④バインダトークン]"
    "Content-type" = "application/json"
}
$Body = @{
    "[⑤署名の部品キー]" = $Headers = @{
        "Authorization" = "Bearer [⑤バインダトークン]"
        "Content-type" = "application/json"
    }
}
$Body = $Body | ConvertTo-Json
$Body = [System.Text.Encoding]::UTF8.GetBytes($Body)
Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .\response_PUTDoc.json
```

```
$Headers = @{
    "Authorization" = "Bearer FK8m_a65m0pFcxJIGCiAi0kAiBrv2v8p"
    "Content-type" = "application/json"
}
```

リクエストボディの項目を指定

SmartDBの新フォーム定義

リクエスト

```

$URI = "[①プロトコル+ホスト名+リソースパス]"
$Method = "[②HTTPメソッド]"
$headers = @{
    "[③認証用のヘッダ]" = "Bearer [④バインダトークン]"
    "Content-type" = "application/json"
}
$body = @{
    "[⑤部署名の部品キー]" = "[⑥部署名の値]"
}
    
```

```

$body = $body | ConvertTo-Json
$body = [System.Text.Encoding]::UTF8.GetBytes($body)
Invoke-WebRequest $URI -Method $Method -Body $body -ContentType application/json -Headers $headers -OutFile PUTDoc.json
    
```

```

$body = @{
    "[⑤部署名の部品キー]" = "[⑥部署名の値]"
}
    
```

```

$body = @{
    "departmentName" = "[⑥部署名の値]"
}
    
```

リクエストボディの項目を指定

取得した文書データ

変更後

項目	値
人物ID	AA0001
氏名	夢野 芸
会社名	株式会社ドリーム・アーツ
部署名	経営企画部
	⋮

リクエスト

```

$URI = "[①プロトコル+ホスト名+リソースパス]"
$Method = "[②HTTPメソッド]"
$headers = @{
    "[③認証用のヘッダ]" = "Bearer [④バインダトークン]"
    "Content-type" = "application/json"
}
$body = @{
    "[⑤部署名の部品キー]" = "[⑥部署名の値]"
}
$body = $body | ConvertTo-Json
[System.Text.Encoding]::UTF8.GetBytes($body)
Invoke-WebRequest $URI -Method $Method -Body $body
    
```

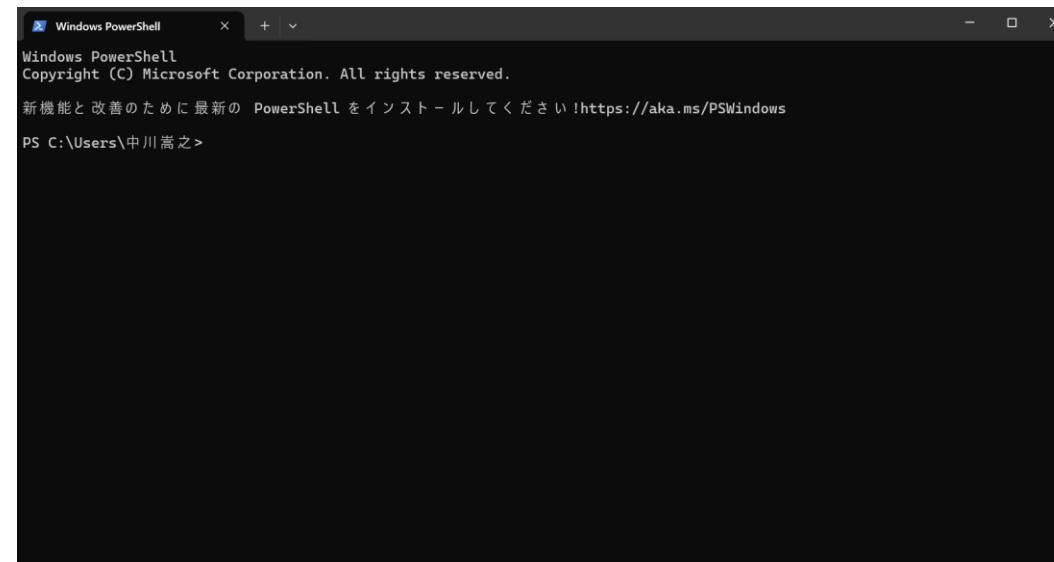
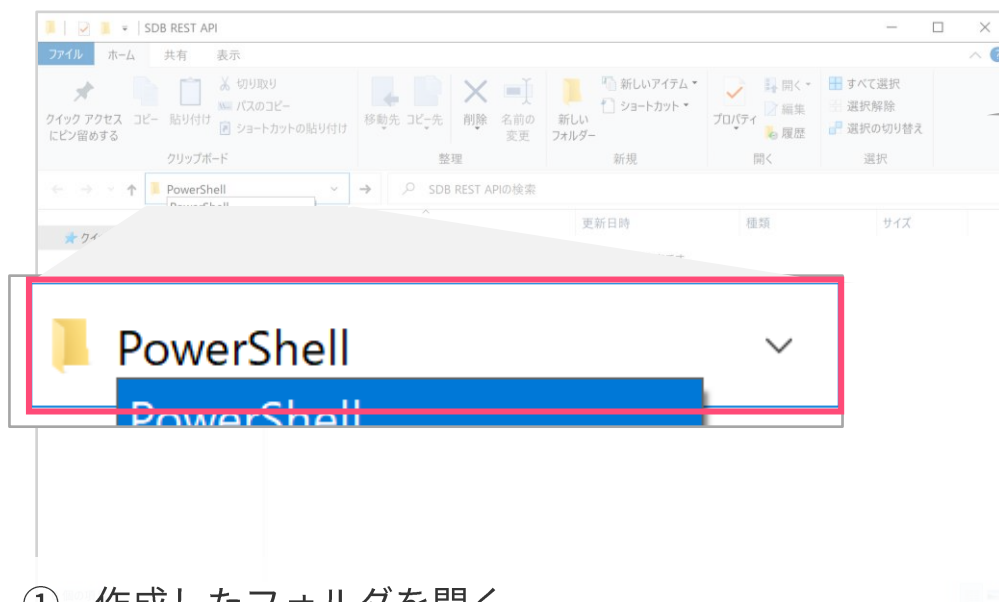
```

$body = @{
    "departmentName" = "[⑥部署名の値]"
}
PUT Doc.json
    
```

```

$body = @{
    "departmentName" = 経営企画部
}
    
```

作成したフォルダからPowerShellを起動



- ① 作成したフォルダを開く
- ② アドレスバーに「PowerShell」と入力する
- ③ Enterキーを押す

※ログファイルの出力先を今回作成したフォルダにするため、フォルダから起動しています。

作成したコードをPowerShellで実行

【記入用】 「文書更新」の実行コード

DreamArts

```
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/documents/1"
$Method = "PUT"
$headers = @{
    "Authorization" = "Bearer FK8m_a65m0pFcxJlGcAi0kAiBrv2v8p"
    "Content-type" = "application/json"
}
$Body = @{
    "departmentName" = "経営企画部"
}
$Body = $Body | ConvertTo-Json
$Body = [System.Text.Encoding]::UTF8.GetBytes($Body)
Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $Body -UseBasicParsing -OutFile .\response_PUTDoc.json
```

- ① 作成したコードをコピーする
- ② PowerShellに貼り付ける
- ③ Enterキーで実行する

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

C:\Users\中川嵩之\Desktop\SDB REST API> $URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/documents/1"
C:\Users\中川嵩之\Desktop\SDB REST API> $Method = "PUT"
C:\Users\中川嵩之\Desktop\SDB REST API> $headers = @{
    "Authorization" = "Bearer FK8m_a65m0pFcxJlGcAi0kAiBrv2v8p"
    "Content-type" = "application/json"
}
C:\Users\中川嵩之\Desktop\SDB REST API> $Body = @{
    "departmentName" = "経営企画部"
}
C:\Users\中川嵩之\Desktop\SDB REST API> $Body = $Body | ConvertTo-Json
C:\Users\中川嵩之\Desktop\SDB REST API> $Body = [System.Text.Encoding]::UTF8.GetBytes($Body)
C:\Users\中川嵩之\Desktop\SDB REST API> Invoke-WebRequest $URI -Method $Method -Headers $headers -Body $Body -UseBasic
Parsing -OutFile .\response_PUTDoc.json
PS C:\Users\中川嵩之\Desktop\SDB REST API>
```

実行結果を確認

成功

失敗

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\中川嵩之\Desktop\SDB REST API> $URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/documents/1"
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Method = "PUT"
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Headers = @{
>>   "Authorization" = "Bearer FK8m_a65m0pFcxlGciAi0kAiBrv2v8p"
>>   "Content-type"="application/json"
>> }
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Body = @{
>>   "departmentName" = "経営企画部"
>> }
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Body = $Body | ConvertTo-Json
PS C:\Users\中川嵩之\Desktop\SDB REST API> $Body = [System.Text.Encoding]::UTF8.GetBytes($Body)
PS C:\Users\中川嵩之\Desktop\SDB REST API> Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .\response_PUTDoc.json
PS C:\Users\中川嵩之\Desktop\SDB REST API>
    
```

エラーが出力されない

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\中川嵩之> $URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13243/documents/1"
PS C:\Users\中川嵩之> $Method = "PUT"
PS C:\Users\中川嵩之> $Headers = @{
>>   "Authorization" = "Bearer FK8m_a65m0pFcxlGciAi0kAiBrv2v8p"
>>   "Content-type"="application/json"
>> }
PS C:\Users\中川嵩之> $Body = @{
>>   "departmentName" = "経営企画部"
>> }
PS C:\Users\中川嵩之> $Body = $Body | ConvertTo-Json
PS C:\Users\中川嵩之> $Body = [System.Text.Encoding]::UTF8.GetBytes($Body)
PS C:\Users\中川嵩之> Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body -UseBasicParsing -OutFile .\response_PUTDoc.json
Invoke-WebRequest : リモート サーバーがエラーを返しました: (401) 許可されていません
発生場所 行:1 文字:1
+ Invoke-WebRequest $URI -Method $Method -Headers $Headers -Body $Body ...
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
PS C:\Users\中川嵩之>
    
```

赤字でエラーが出力

変更前

一覧 編集 再利用 更新履歴 削除

🔍 ブックマーク

文書タイトル	夢野芸太郎		
文書番号	1	更新	10:35 中川 嵩之

顧客マスタ

■ 登録情報

登録日	2024/02/29
人物ID	AA0001
氏名	夢野芸太郎
氏名 (フリガナ)	ユメノゲイタロウ
会社名	株式会社ドリーム・アーツ
部署名	営業部

■ 担当営業者名

担当営業者	中川 嵩之
-------	-------

🔍 ブックマーク

一覧 編集 再利用 更新履歴 削除

変更後

一覧 編集 再利用 更新履歴 削除

🔍 ブックマーク

文書タイトル	夢野芸太郎		
文書番号	1	更新	10:36 中川 嵩之

顧客マスタ

■ 登録情報

登録日	2024/02/29
人物ID	AA0001
氏名	夢野芸太郎
氏名 (フリガナ)	ユメノゲイタロウ
会社名	株式会社ドリーム・アーツ
部署名	経営企画部

■ 担当営業者名

担当営業者	中川 嵩之
-------	-------

🔍 ブックマーク

一覧 編集 再利用 更新履歴 削除

1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

1. SmartDB REST APIの利用には、**バインダトークン**が必要
2. **RestDocV3**、**SmartDB環境情報**を参照してリクエスト作成
3. SmartDB REST APIを**組み合わせる**ことで業務を実現
4. 最もよく使うSmartDB REST APIは下表の通り

カテゴリ	REST API
バインダ / 文書	文書詳細情報取得 指定したビューの文書一覧取得 文書一覧のCSV出力 文書新規登録 文書更新 文書削除 文書一括操作 CSV入力（新規） CSV入力（更新）
業務プロセス	業務プロセス開始 アクティビティ実施 文書情報に基づいて実行中アクティビティ情報取得



1. 前回の復習
2. SmartDB REST APIの始め方
3. よく使うSmartDB REST API
4. ハンズオン
5. まとめ
6. 次回予告

No.	実施概要	詳細内容	推奨者
1	REST API 入門編	REST API概説 実践：PowerShellでREST APIを実行してみよう	SmartDB利用経験のある方
本日	SmartDB REST API アプリ操作基本編	SmartDB REST APIの基本的な使い方 よく使う基本的なAPIの解説と実践	1の参加者、もしくは1の内容を理解している方
	SmartDB REST API アプリ操作応用編	SmartDB REST APIの応用的な使い方 複数のREST APIを用いた業務適用の解説と実践	2の参加者、もしくは2の内容を理解している方
4	SmartDB REST API アカウントマスタ連携編	アカウントAPIの使い方 アカウントAPIを用いた業務適用の解説と実践	1の参加者、もしくは1の内容を理解している方
5	SmartDB REST API 業務・性能・セキュリティ編	業務影響を軽減するための設計 SDBの性能を考慮した設計・運用	3の参加者、もしくは3の内容を理解している方 または、 4の参加者、もしくは4の内容を理解している方
6	SmartDB REST API システム連携設計編	SmartDBを含む複数システム連携の全体設計 SDB起点の実行方法：Webhook、定期バッチ処理	5の参加者、もしくは5の内容を理解している方

※上記の内容は変更される場合がございます

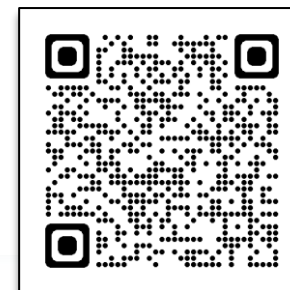
お知らせ

資料や動画は、後日コミュニティサイト「スマラジ！ルーム」へ掲載します。

本イベントに関するご質問も受け付けますのでお気軽に投稿ください。

<https://cs.support-smartdb.com/hc/ja/community/topics/5423952120601>

※閲覧にはサポートサイトへのログインが必要です。



スマラジ！ルーム

新規投稿

過去動画・資料を公開中

フォローする

すべて表示 最新の投稿で並べ替え

10月18日（火）開催：スマラジ！「SmartDB」のデータの活用を広げる「業務ダッシュボード」とは
志保 江森 · 2か月前

10月5日（水）開催：スマラジ！評価式を使って「SmartDB」の活用幅を広げよう
志保 江森 · 2か月前

8月10日（水）開催：スマラジ！学ぶ！SmartDB×他システム連携の第一歩
志保 江森 · 4か月前

コミュニティサイトでは、DAや他ユーザーさんへの質問が可能です。

気になったことがあれば、お気軽に投稿ください。



SmartDBサポートサイト > コミュニティ > 雑談・つぶやき

非推奨のログイン方法だけど・・・SmartDB REST APIをExcel VBAで試した話

あおさん (趣味C#プログラマー) **Great supporter** **First post**

前回に続いて、API関連の話を書いてみます。
 但し、今回のログイン方法は公式に**非推奨**であり、セキュリティ的によろしくなく、またいつ使えなくなってもおかしくないのをご注意ください。
 また、複雑になるのでVBAのコードは書きません。

まず、最新のAPI (V3) を確認してみます。

SmartDB REST API (support-dreamarts.com)

セッションについての説明の3つ目を見ます

- ・ `post /session` を実行して、セッションを作成する。※非推奨

ID/PASSWORDを指定してsession APIを実行してセッションを作成します。
 ここで作成したセッションを利用して、その他のREST APIを実行します。バッチ処理などを特定ユーザで実行したい場合があります。
 プログラム中にユーザのID/パスワードを記述する必要があるため、セキュリティの観点からも非推奨とします。

正直この説明が全てです。

バインダAPIオプション契約前のテストぐらいには使えるかもしれませんが、セキュリティ的に本運用には使えません。

まず、この方法がまだ使えるのか確認します。

お手軽にWindowsのコマンドプロンプトからcurlでやってみます。

ログインが、

サイト内

SmartDBサポートサイト > コミュニティ > 一般Q&A

グラフダイアログのカラー設定

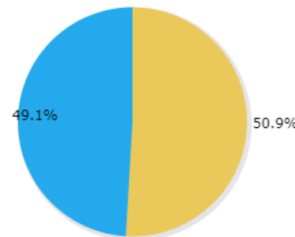
のほほん **Great supporter** **Nice multiple posts**

2023年09

皆さん、グラフダイアログって利用されていますか？
 簡易なグラフなら便利かな？と思って設定していたのですが
 値の大きい順に色設定されてしまう事に気が付きました。
 これって、あたり前の事なのでしょうか(´Д`)

車の運転前と運転後にそれぞれ登録するバインダがあります。
 運転前の登録数と運転後の登録数は原則イコールになる想定なのですが
 明らかに登録割合のおかしい人には注意をしようと思ってグラフを表示しました。

●Aさん



運転前後	部品件数	比率
運転後	55	50.9%
運転前	53	49.1%
合計	108	100.0%

運転前と運転後ではほぼ同じなのを、よく

フォローする

SmartDBサポートサイト > コミュニティ > What's New !

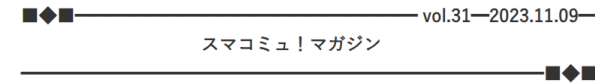
202

スマコミュ！マガジン_vol.31～10/18デジタルの民主化DAYを開催しました！～

フォローする

廣瀬 璃奈 **DreamArts**

2023年11月09日 14:14



こんにちは！ドリーム・アーツの廣瀬です。

涼くなったかと思えば急に気温が上がったりと、なかなか不安定な時期が続いていますね。
 私は週末に少し遠出をして紅葉を見に行きました🍁木々の葉っぱの色が変わっていく瞬間は、季節の変化を感じます。
 皆さんが季節の変わり目を感じる瞬間はいつでしょうか？

今月もSmartDBお役立ち情報を配信していきます。

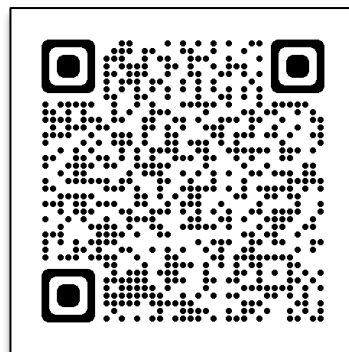
—本日のお役立ち情報—

1. 10/18デジタルの民主化DAYを開催しました！
2. 10月のコミュニティサイト人気投稿ランキング
3. SmartDBのレビューを投稿してAmazonギフト券をゲット！

SmartDB認定制度

SmartDB Certified Specialist

SmartDBによる
業務デジタル化の習熟度・スキルを
個人へ認定するプログラム



詳細はこちら ▶

<https://hibiki.dreamarts.co.jp/smartdb/scs/>



すでに、多くの認定者が誕生！
デジタルの民主化を推進する企業が続々受験されています



※弊社サイトより抜粋

【3月31日まで】 コミュニティ参加者限定

ITreviewへのアンケート掲載でもれなくプレゼント！

Amazonギフト券 **3,000**円分



×



キャンペーンコードを必ずご入力ください

AUQBMHOY

※「キャンペーン要項」を必ずご確認ください



無くなり
次第終了

Step.1



ITreviewに
会員登録をしよう

Step.2



レビュー・口コミ
を書こう

Step.3

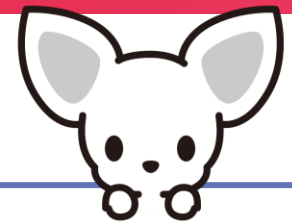


掲載されると...

Step.4



Amazonギフト券GET!!



[キャンペーン専用ページURL](#)

DreamArts

<https://www.dreamarts.co.jp/>

